Large (Brain) Graph Matching via Fast Approximate Quadratic Programming

Joshua T. Vogelstein, John M. Conroy, Louis J. Podrazik, Steven G. Kratzer, Eric T. Harley, Donniell E. Fishkind, R. Jacob Vogelstein, and Carey E. Priebe

November 16, 2012

Abstract

Graph matching (GM)—the process of finding an optimal permutation of the vertices of one graph to minimize adjacency disagreements with the vertices of another—is rapidly becoming an increasingly important computational problem, arising in fields ranging from machine vision to neuroscience. Because GM is $\mathcal{NP}$-hard, exact algorithms are unsuitable for today's large graphs (with ≥ 1000 vertices). Approximate algorithms necessarily employ a accuracy/efficiency trade-off. We developed a fast approximate quadratic assignment algorithm (`FAQ`). `FAQ` scales cubicly with the number of vertices, similar to other approximate GM algorithms. Our formulation, however, achieves a lower objective function value than all previously proposed algorithms on over 93% of the QAPLIB benchmark problems. Moreover, our algorithm runs faster than the second best algorithm on over 80% of the benchmarks. We therefore implement our algorithm on a dataset of increasing interest to the neurobiology community: the brain-graph (connectome) of the C. elegans. `FAQ` solves this problem perfectly, and no other algorithm we tried was successful. Future work will hopefully improve upon the cubic complexity of `FAQ` to be able to match mammalian brain-graphs, with millions or billions of vertices.

1 Introduction

Graph matching—the process of finding an optimal permutation of the vertices of one graph to minimize adjacency disagreements with the vertices of another—is a famously computationally daunting problem (see, for example, "Thirty Years of Graph Matching in Pattern Recognition" [1]). Specifically, graph matching is an $\mathcal{NP}$-hard problem, in particular, we do not know whether a polynomial time algorithm can solve it in worst case scenarios [2]. As such, performance of graph matching algorithms are usually evaluated on graphs with $\approx$10 vertices, or at maximum $\approx$100 (see [3] for a description of the standard set of benchmarks). Yet, it is increasingly popular to represent large data sets by a graph, and thus increasingly desirable to consider matching large graphs.

The motivating application for this work is *brain-graph matching*. A brain-graph (aka, a connectome) is a graph for which vertices represent (collections of) neurons and edges represent connections between them [4, 5]. Via Magnetic resonance (MR) imaging, one can image the whole brain and estimate connectivity across voxels, yielding a voxelwise connectome with up to $\approx 10^6$ vertices and $\approx 10^9$ edges [6]. Comparing brains is an important step for many neurobiological inference tasks. For example, it is becoming increasingly popular to diagnose neurological diseases via comparing brain images [7]. To date, however, these comparisons have largely rested on anatomical (e.g., shape) comparisons, not graph comparisons. This is despite the widely held doctrine that many psychiatric disorders are fundamentally "connectopathies", that is, disorders of the connections of the brain [8–11]. Currently available tests for connectopic explanation of psychiatric disorders hedge upon first choosing some number of graph invariants to compare across populations. The graph invariant approach to classifying is both theoretically and practically inferior to comparing whole graphs via matching [12].

More generally, state-of-the-art inference procedures for essentially any decision-theoretic or inference task follow from constructing interpoint dissimilarity matrices, or graphs [13]. Thus, we believe that graph matching of large graphs will become a fundamental subroutine of many statistical inference pipelines operating on graphs. Because the number of vertices of these graphs is so large, exact matching is intractable.

Instead, we require inexact matching algorithms (also called "heuristics") that will scale polynomially or even linear [1]. In developing such algorithms, there is an inherent accuracy/efficiency trade-off. If an algorithm outperforms another on both dimensions, it is clearly preferential.

The remainder of this paper is organized as follows. Section 2 formally defines the "graph matching" problem, and Section 3 makes the connection between graph matching and quadratic assignment problems (QAPs). Section 4 describes our algorithm, a Fast Approximate QAP (`FAQ`). Section 5 provides a number of theoretical and empirical results, comparing our algorithm to previous state-of-the-art algorithms in terms of both computational efficiency and objective function value. We conclude with a discussion in Section 6.

2 Graph Matching

A labeled graph $G = (\mathcal{V}, \mathcal{E})$ consists of a vertex set $\mathcal{V}$, where $|\mathcal{V}| = n$ is number of vertices, and an edge set $\mathcal{E}$. Note that we are not restricting our formulation to be directed or exclude self-loops. Given a pair of graphs, $G_A = (\mathcal{V}_A, \mathcal{E}_A)$ and $G_B = (\mathcal{V}_B, \mathcal{E}_B)$, where $|\mathcal{V}_A| = |\mathcal{V}_B| = n$, let Π be the set of permutation functions (bijections), $\Pi = \{\pi \colon \mathcal{V}_A \to \mathcal{V}_B\}$. Now consider the following two closely related problems:

- **Graph Isomorphism (GI):** Does there exist a $\pi \in \Pi$ such that $(u, v) \in \mathcal{E}_A$ if and only if $(\pi(u), \pi(v)) \in \mathcal{E}_B$.
- **Graph Matching (GM):** Which $\pi \in \Pi$ minimizes adjacency disagreements between $\mathcal{E}_A$ and the permuted $\mathcal{E}_B$?

Both GI and GM are computationally difficult. GM is at least as hard as GI, since solving GM also solves GI, but not vice versa. It is not known whether GI is in complexity class $\mathcal{P}$ [14]. In fact, GI is one of the few problems for which, if $\mathcal{P} \neq \mathcal{NP}$, then GI might reside in an intermediate complexity class called $\mathcal{GI}$-complete. GM, however, is known to be $\mathcal{NP}$-hard. Yet, for large classes of GI and GM problems, linear or polynomial time algorithms are available [15]. Moreover, at worst, it is clear that GI is only "moderately exponential," for example, $\mathcal{O}(\exp\{n^{1/2+o(1)}\})$ [16]. Unfortunately, even when linear or polynomial time GI or GM algorithms are available for special cases of graphs, the constants are often unbearably large. For example, if all vertices have degree less than k, there is a linear time algorithm for GI. However, the hidden constant in this algorithm is $512k^3!$ (yes, that is a factorial!) [17].

Because we are interested in solving GM for graphs with $\dot{\approx}10^6$ or more vertices, exact GM solutions will be computationally intractable. As such, we develop a fast approximate graph matching algorithm. Our approach is based on formulating GM as a quadratic assignment problem (QAP).

3 Graph Matching as a QAP

Graph matching can be formulated as a quadratic assignment problem (QAP). Let $A = (a_{uv}) \in \{0, 1\}^{n \times n}$ and $B = (b_{uv}) \in \{0, 1\}^{n \times n}$ correspond to the adjacency matrix representations of two graphs that we desire to match. That is, let $a_{uv} = 1$ if and only if $(u, v) \in \mathcal{E}_A$, and similarly for b_{uv}. Moreover, let $\mathcal{P}$ be the set of $n \times n$ *permutation matrices* $\mathcal{P} = \{P : P^\mathsf{T}\mathbf{1} = P\mathbf{1} = \mathbf{1}, P \in \{0, 1\}^{n \times n}\}$, where $\mathbf{1}$ is an n-dimensional column vector. We therefore have the following problem:

$$
\begin{aligned}
\text{(QAP)} \quad & \operatorname*{argmin}_{\pi \in \Pi} \sum_{i,j \in [n]} (a_{ij} - b_{\pi(i)\pi(j)})^2 = && \text{(1a)}\\
& \operatorname*{argmin}_{P \in \mathcal{P}} \left\| A - PBP^{\mathsf{T}} \right\|_F = && \text{(1b)}\\
& \operatorname*{argmin}_{P \in \mathcal{P}} tr(A - PBP^{\mathsf{T}})^{\mathsf{T}}(A - PBP^{\mathsf{T}}) = && \text{(1c)}\\
& \operatorname*{argmin}_{P \in \mathcal{P}} -tr(BP^{\mathsf{T}}AP), && \text{(1d)}
\end{aligned}
$$

where the last equality follows from dropping terms that cancel because P is a permutation matrix. Note that the above algebraic formulation of GM facilitates generalizing the original problem statement. In particular, one can now search for the permutation that minimizes a particular objective function, $f(P) = -tr(BP^{\mathsf{T}}AP)$. Moreover, it is natural to consider "weighted graph matching" problems, in which each edge is associated with a weight, $a_{uv} \in \mathbb{R}$.

Our approach follows from relaxing the above binary constraints to be non-negative constraints, yielding a quadratic program with *linear* constraints. Thus, the feasible region expands to the convex hull of the permutation matrices: the doubly stochastic matrices, $\mathcal{D} = \{P : P^{\mathsf{T}}\mathbf{1} = P\mathbf{1} = \mathbf{1}, P \succeq 0\}$, where $\succeq$ indicates an element-wise inequality:

$$
\begin{aligned}
\text{(rQAP)} \quad & \operatorname*{minimize}_{P} - tr(BP^{\mathsf{T}}AP) && \text{(2a)}\\
& \text{subject to } P \in \mathcal{D}. && \text{(2b)}
\end{aligned}
$$

rQAP—the above relaxed Quadratic Assignment Problem—is quadratic but not necessarily convex, because the Hessian of its objective function is not necessarily positive definite:

$$
\nabla^2 f(P) = -B \otimes A - B^{\mathsf{T}} \otimes A^{\mathsf{T}}, \tag{3}
$$

where $\otimes$ indicates the Kronecker product. This means that the solution space will potentially be multimodal, making initialization important. With this in mind, below, we describe an algorithm to find a local optimum of rQAP.

4 Fast Approximate Quadratic Assignment Problem Algorithm

Our algorithm, called `FAQ`, has three components:

A. Choose a suitable initial position.

B. Find a local solution to rQAP.

C. Project onto the set of permutation matrices.

Below, we provide details for each component.

A: Find a suitable initial position. While any doubly stochastic matrix would be a feasible initial point, we choose the "flat doubly stochastic matrix," $J = \mathbf{1} \cdot \mathbf{1}^{\mathsf{T}}/n$, which is the barycenter of the feasible region.

B: Find a local solution to rQAP. As mentioned above, rQAP is a quadratic problem with linear constraints. A number of off-the-shelf algorithms are readily available for finding local optima in such problems. We utilize the Frank-Wolfe algorithm (`FW`), a successive linear programing problem originally devised to solve quadratic problems with linear constraints [18, 19].

Although `FW` is a relatively standard solver, especially as a subroutine for QAP algorithms [20], below we provide a detailed view of applying `FW` to rQAP. Given an initial position, $P^{(0)}$, iterate the following four steps.

Step 1: Compute the gradient $\nabla f(P^{(i)})$*:* The gradient f with respect to P is given by

$$\nabla f(P^{(i)}) = -AP^{(i)}B^{\mathsf{T}} - A^{\mathsf{T}}P^{(i)}B. \tag{4}$$

Step 2: Compute a new putative point $\widetilde{P}^{(i+1)}$*:* The new putative point is given by the argument that minimizes a first-order Taylor series approximation to $f(P)$ around the current estimate, $P^{(i)}$. The first-order Taylor series approximation to $f(P)$ is given by

$$\widetilde{f}^{(i)}(P) \overset{\triangle}{=} f(P^{(i)}) + \nabla f(P^{(i)})^{\mathsf{T}}(P - P^{(i)}). \tag{5}$$

Thus, Step 2 of `FW` is

$$\widetilde{P}^{(i+1)} = \operatorname*{argmin}_{P\in\mathcal{D}} f(P^{(i)}) + \nabla f(P^{(i)})^{\mathsf{T}}(P - P^{(i)}) \tag{6a}$$

$$= \operatorname*{argmin}_{P\in\mathcal{D}} \nabla f(P^{(i)})^{\mathsf{T}}P. \tag{6b}$$

As it turns out, Eq. (6b) can be solved as a *Linear Assignment Problem* (LAP). The details of LAPs are well known [21], so we relegate them to the appendix. Suffice it to say here, LAPs can be solved via the "Hungarian Algorithm", named after three Hungarian mathematicians [22–24]. Modern variants of the Hungarian algorithm are cubic in n, that is, $\mathcal{O}(n^3)$, or even faster in the case of sparse or otherwise structured graphs [21,25]. The $\mathcal{O}(n^3)$ computational complexity of `FW` was the primary motivating factor for utilizing `FW`; generic linear programs can require up to $\mathcal{O}(n^7)$.

Step 3: Compute the step size $\alpha^{(i)}$ Given $\widetilde{P}^{(i+1)}$, the new point is given maximizing the *original* optimization problem, rQAP, along the line segment from $P^{(i)}$ to $\widetilde{P}^{(i+1)}$ in $\mathcal{D}$.

$$\alpha^{(i)} = \operatorname*{argmin}_{\alpha\in[0,1]} f(P^{(i)} + \alpha^{(i)}\widetilde{P}^{(i)}). \tag{7}$$

This can be performed exactly, because f is a quadratic function.

Step 4: Update $P^{(i)}$ Finally, the new estimated doubly stochastic matrix is given by

$$P^{(i+1)} = P^{(i)} + \alpha^{(i)}\widetilde{P}^{(i+1)}. \tag{8}$$

Stopping criteria Steps 1–4 are iterated until some stopping criterion is met (computational budget limits, $P^{(i)}$ stops changing much, or $\nabla f(P^{(i)})$ is close to zero). These four steps collectively comprise the Frank-Wolfe algorithm for solving rQAP.

C: Project onto the set of permutation matrices. Let $\widehat{D}$ be the doubly stochastic matrix resulting from the final iteration of `FW`. We project $\widehat{D}$ onto the set of permutation matrices, yielding

$$\widehat{P} = \operatorname*{argmin}_{P\in\mathcal{P}} -\langle \widehat{D}, P\rangle, \tag{9}$$

where $\langle \cdot,\cdot\rangle$ is the usual Euclidean inner product, i.e., $\langle X, Y\rangle \overset{\triangle}{=} tr(X^{\mathsf{T}}Y) = \sum_{ij} x_{ij}y_{ij}$. Note that Eq. (9) is a LAP (again, see appendix for details).

5 Results

5.1 Algorithm Complexity and leading constants

As mentioned above, GM is computationally difficult; even those special cases for which polynomial time algorithms are available, the leading constants are intractably large for all but the simplest cases. We therefore determined the average complexity of our algorithm and the leading constants. The primary computational bottleneck of FAQ is solving the LAP as a subroutine. We use the Jonker and Volgenant version of the Hungarian algorithm [21,25], which is known to scale cubically in the number of vertices. Figure 1 suggests that FAQ is not just cubic in time, but also has very small leading constants ($\dot{\approx}10^{-9}$ seconds), making using this algorithm feasible for even reasonably large graphs. Note that the other state-of-the-art approximate graph matching algorithms also have cubic or worse time complexity in the number of vertices. We will describe these other algorithms and their time complexity in greater detail below.

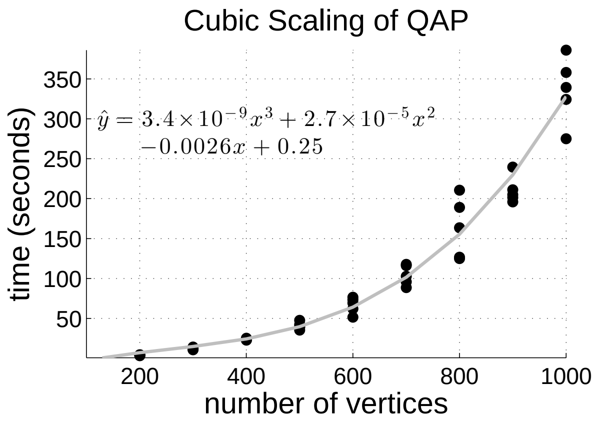

Figure 1: Running time of FAQ as function of number of vertices. Data was sampled from an Erdös-Rényi model with $p = log(n)/n$. Each dot represents a single simulation, with 100 simulations per n. The solid line is the best fit cubic function. Note the leading constant is $\dot{\approx}10^{-9}$ seconds. FAQ finds the optimal objective function value in every simulation.

5.2 QAP Benchmark Accuracy

Having demonstrated empirically FAQ has cubic time complexity, we next decided to evaluate its accuracy on a suite of standard benchmarks. More specifically, QAPLIB is a library of 137 quadratic assignment problems, ranging in size from 10 to 256 vertices [3]. Recent graph matching papers typically evaluate the performance of their algorithm on 16 of the benchmarks that are known to be "particularly difficult" [26,27]. We compare the results of FAQ to the results of four other state-of-the-art graph matching algorithms: (1) the PATH algorithm, which solves a path between a convex and concave relaxation of QAP [26], (2) QCV which is the convex relaxation used to initialize the PATH algorithm, (3) the RANK algorithm [28], which uses a spectral decomposition, and (4) the Umeyama algorithm (denoted by U henceforth), which also uses a spectral decomposition [29]. We chose these four algorithms to compare because the code is freely available from the graphm package [26]. Figure 2 plots the logarithm (base 10, here and elsewhere) of the relative accuracy, that is, $\log_{10}(\hat{f}_{FAQ}/\hat{f}_X)$, for $X \in \{$PATH, QCV, RANK, U, all$\}$, where all is just the best performer of all the non FAQ algorithms. Clearly, FAQ does significantly better than all the other algorithms, outperforming all of them on $\approx 94\%$ of the problems, often by nearly an order of magnitude in terms of relative error.

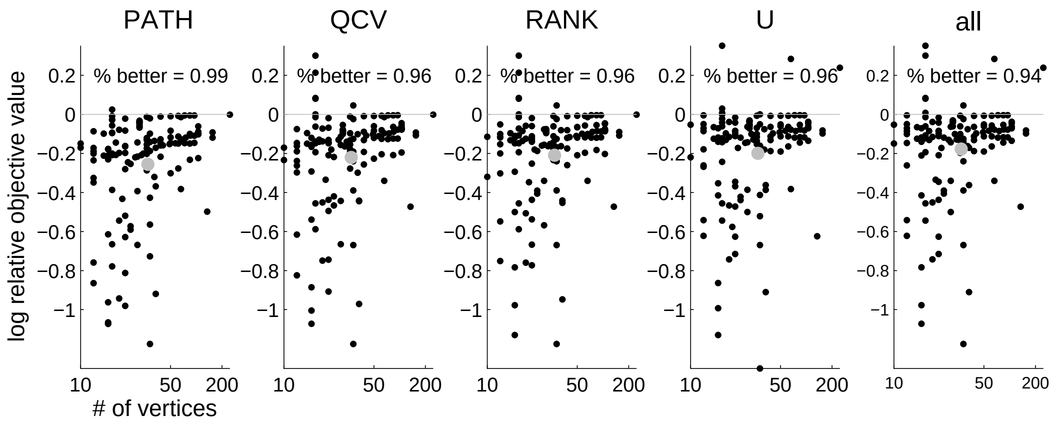

Figure 2: Relative accuracy—defined to be $\log_{10}(\hat{f}_{FAQ}/\hat{f}_X)$—of all the four algorithms compared with FAQ. Note that FAQ is better than all the other algorithms on $\approx 94\%$ of the benchmarks. The abscissa is the log number of vertices. The gray dot indicates the mean improvement of FAQ over the other algorithms.

5.3 QAP Benchmark Efficiency

As we mentioned in the introduction, the quality of an *approximate* algorithm depends not just on its accuracy, but also its efficiency. Therefore, we compare the wall time of each of the five algorithms on all 137 benchmarks in Figure 3. We fit an iteratively weighted least squares linear regression function (Matlab's `robustfit`) to regress the logarithm of time (in seconds) onto the logarithm of the number of vertices. The numbers beside the lines indicate the slopes of the regression functions. The PATH algorithm has the worst slope. QCV and FAQ have nearly identical slopes, which makes sense, given that the are solving very similar objective functions. Similarly, RANK and U have very similar slopes; they are both using spectral approaches. Note, however, that although the slope of RANk and U are smaller than that of FAQ, they both seem to be super linear on this log-log plot, suggesting that as the number of vertices increases, their compute time might exceed that of the other algorithms.

5.4 QAP Benchmark Accuracy/Efficiency Trade-off

In the PATH, the authors demonstrated that PATH outperformed QCV and U on a variety of simulated and real examples in terms of objective function [26]. If FAQ yields a lower objective function value than FAQ, and is faster, then it clearly is superior to PATH. Figure 4 compares the performance of FAQ with PATH along both dimensions of performance—accuracy and efficiency—for all 137 benchmarks in the QAPLIB library. The right panel indicates that FAQ is both more accurate and more efficient on 80% of the problems.

5.5 QAP Directed Benchmarks

While this manuscript was under review, Liu et al. [30] proposed a modification of the PATH algorithm that adjusted PATH to be more appropriate for directed graphs, as the theory motivating the development of PATH relied upon the graphs being undirected. FAQ, on the other hand, does not depend on the graphs being simple; rather, directed or weighted graphs are both unproblematic. Liu et al. compare the performance of their algorithm (EPATH) with U, QCV, and GRAD on the set of 16 particularly difficult directed benchmarks from QAPLIB. The EPATH algorithm achieves at least as low objective value as the other algorithms on 15 of 16 benchmarks. Our algorithm, FAQ, always gets the best of the five algorithms. Table 1 shows the numerical results comparing FAQ to EPATH and GRAD, which sometimes did better than EPATH. Note that some of the algorithms achieve the absolute minimum on some benchmarks.

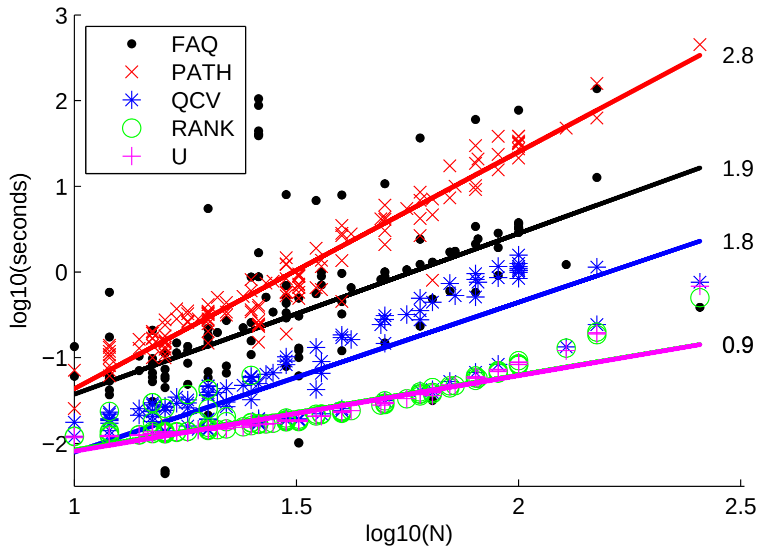

Figure 3: Absolute wall time for running each of the five algorithms on all 137 benchmarks. We fit a line on this log-log plot for each algorithm; the slope is displayed beside each line. The `FAQ` slope is much better than the `PATH` slope, and worse than the others. Note, however, the time for `RANK` and `U` appears to be superlinear on this log-log plot, suggesting that perhaps as the number of vertices increases, `PATH` might be faster.

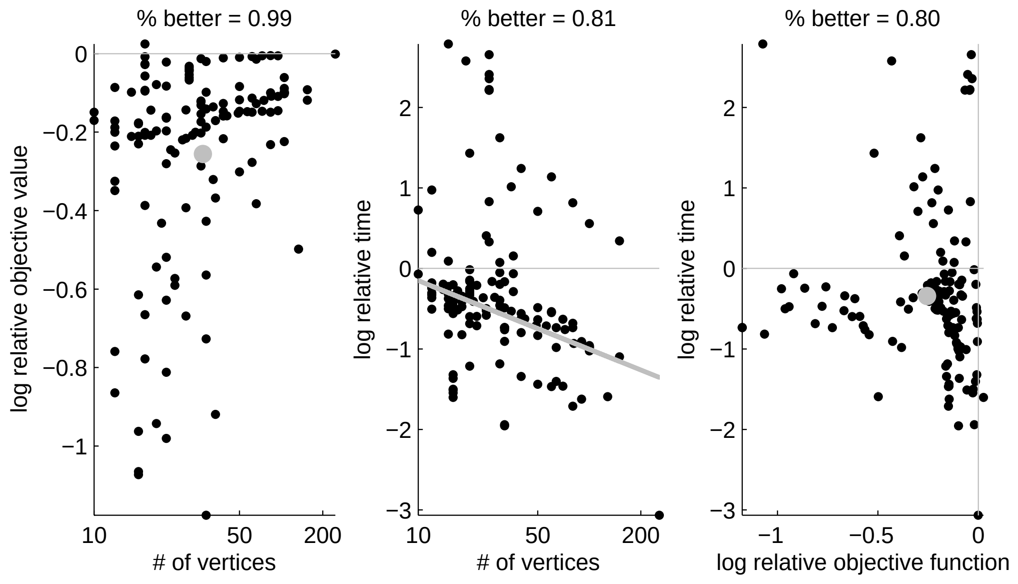

Figure 4: Comparison of `FAQ` with `PATH` in terms of both accuracy and efficiency. The left panel is the same as the left panel of Figure 2. The middle plots the relative wall time of `FAQ` to `PATH` as a function of the number of vertices, also on a log-log scale. The gray line is the best fit slope on this plot, suggesting that `FAQ` is getting exponentially faster than `PATH` as the number of vertices gets larger. Finally, the right panel plots log relative time versus log relative objective function value, demonstrating that `FAQ` outperforms `PATH` on both dimensions on 80% of the benchmarks.

Table 1: Comparison of FAQ with optimal objective function value and previous state-of-the-art for directed graphs. The best (lowest) value is in **bold**. Asterisks indicate achievement of the global minimum. The number of vertices for each problem is the number in its name (second column).

#	Problem	Optimal	FAQ	EPATH	GRAD
1	lipa20a	3683	**3791**	3885	3909
2	lipa20b	27076	**27076***	32081	**27076***
3	lipa30a	13178	**13571**	13577	13668
4	lipa30b	151426	**151426***	**151426***	**151426***
5	lipa40a	31538	**32109**	32247	32590
6	lipa40b	476581	**476581***	**476581***	**476581***
7	lipa50a	62093	**62962**	63339	63730
8	lipa50b	1210244	**1210244***	**1210244***	**1210244***
9	lipa60a	107218	**108488**	109168	109809
10	lipa60b	2520135	**2520135***	**2520135***	**2520135***
11	lipa70a	169755	**171820**	172200	173172
12	lipa70b	4603200	**4603200***	**4603200***	**4603200***
13	lipa80a	253195	**256073**	256601	258218
14	lipa80b	7763962	**7763962***	**7763962***	**7763962***
15	lipa90a	360630	**363937**	365233	366743
16	lipa90b	12490441	**12490441***	**12490441***	**12490441***

5.6 Theoretical properties of FAQ

Although we have no theorems proving error bounds on FAQ relative to the optimal solution, we do have a number of theoretical results to buttress the numerical ones. As mentioned above in §5.1, FAQ is a cubic-time algorithm, as its computational bottleneck is solving the LAPs, which are solved in $\mathcal{O}(n^3)$ by various algorithms collectively referred to as the "Hungarian algorithm" [21,25].

In addition to guarantees on computational time, we have several guarantees on performance. Consider Eq. (1), and note that Eq. (1d) follows from (1c) by dropping cross-terms that fall out of the optimization because P is constrained to be a permutation matrix. FAQ tries to solve a relaxation of Eq. (1d), specifically relaxing the permutation matrix constraints to their convex hull. The resulting objective function, Eq. (2), however, is quadratic but not convex. A different formulation and relaxation of QAP yields a convex objective function. Specifically,

$$\underset{P}{\text{minimize}} \, \|AP - PB\|_F \tag{10a}$$

$$\text{subject to } P \in \mathcal{P}, \tag{10b}$$

is also $\mathcal{NP}$-hard, but upon relaxing (10b) to the doubly stochastic matrices, the result is convex [31]. Thus, a linear programming algorithm is guaranteed to find the optimal solution to Eq. (10). This relaxation is in fact the objective function that QCV is solving.

If we had relaxed the constraints prior to canceling those terms, this equality would not follow. This leads us to wonder in which circumstances are the objective functions of QAP and rQAP equal. The following lemma clarifies:

Lemma 1. *If A and B are the adjacency matrices of simple graphs (symmetric, hollow, and binary) that are isomorphic to one another, then the minimum of rQAP is equal to the minimum of QAP.*

Proof. Because any feasible solution to QAP is also a feasible solution to rQAP, we must only show that the optimal objective function value to rQAP can be no better than the optimal objective function value of QAP. Let $A = PBP^{\mathsf{T}}$, so that $\langle A, PBP^{\mathsf{T}} \rangle = 2m$, where m is the number of edges in A. If rQAP could achieve a lower objective value, then it must be that there exists a $D \in \mathcal{D}$ such that $\langle A, DBD^{\mathsf{T}} \rangle > \langle A, PBP^{\mathsf{T}} \rangle = 2m$ (remember that we are minimizing the negative Euclidean inner product). For that to be the case, it must be that $(DBD^{\mathsf{T}})_{ij} \geq 1$ for some (u, v). That this is not so may be seen by the submultiplicativity of the norm induced by the ℓ_∞ norm: $\|Dx\|_\infty \leq \|D\|_{\infty,\infty} \|x\|_\infty$. Applying this twice (once for each doubly stochastic matrix multiplication) yields our result. □

5.7 Multiple Restarts

Although FAQ outperformed all other algorithms on nearly every benchmark, that FAQ was not always the best was annoying to us. We therefore utilized the non-convexity of rQAP is as a feature, although it can equally well be regarded as a bug (because rQAP is non-convex so the solution found by FAQ depends on the initial condition). We can utilize the non-convexity as a feature, however, whenever (i) we have some reason to believe that better solutions exist (many algorithms efficiently compute relatively tight lower bounds [32]), and (ii) we can efficiently search the space of initial conditions. Although we lack any supporting theory of optimality, we do know how to sample feasible starting points. Specifically, we desire that our starting points are "near" the flat matrix, and satisfy the conditions. Therefore, we sample $K \in \mathcal{D}$, a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [33], and let our initial guess be $P^{(0)} = (J + K)/2$, where J is the doubly flat matrix. We can therefore use any number of restarts with this approach. Fixing the number of restarts, we still have a cubic time algorithm, although the constants change.

Table 2 shows the performance of running FAQ 3 and 100 times, reporting only the best result (indicated by $\texttt{FAQ}_3$ and $\texttt{FAQ}_{100}$, respectively), and comparing it to the best performing result of the five algorithms (running only FAQ once). Note that we only consider the 16 particularly difficult benchmarks for this evaluation. FAQ only required three restarts to outperform all other approximate algorithms on all 16 of 16 difficult benchmarks. Moreover, after 100 restarts, FAQ finds the absolute minimum on 3 of the 16 benchmarks; none of the other algorithms ever achieved the absolute minimum on any of these benchmarks.

5.8 Brain-Graph Matching

A "chemical connectome" is a brain-graph in which vertices correspond to (collections of) neurons, and edges correspond to chemical synapses between them. The *Caenorhabditis elegans* (*C. elegans*) is a small worm (nematode) with 302 labeled vertices. We consider the subgraph with 279 somatic neurons that form edges with other neurons [34, 35].

Because pairs of neurons sometimes have multiple synapses between them, and they are directed, the chemical connectome of *C. elegans* may be thought of as a weighted directed graph. We therefore conducted the following synthetic experiments. Let $A_{ij} \in \{0, 1, 2, \ldots\}$ be the number of synapses from neuron i to neuron j, and let $A = \{A_{ij}\}_{i,j \in [279]}$. To generate synthetic data, we let $B^{(k)} = Q^{(k)} A {Q^{(k)}}^{\mathsf{T}}$, for some $Q^{(k)}$ chosen uniformly at random from $\mathcal{P}$, effectively shuffling the vertex labels of the connectome. Then, we try to graph match A to $B^{(k)}$, for $k = 1, 2, \ldots, 1000$, that is, we repeat the experiment 1000 times. We define accuracy as the fraction of vertices correctly assigned. We always start with the doubly flat matrix.

Figure 5 displays the results of FAQ along with U, QCV, and PATH. The left panel indicates that FAQ *always* found the optimal solution for the chemical connectome, whereas none of the other algorithms *ever* found the optimal solution. The right panel compares the wall time of the various algorithms, running on an 2.2 GHz Apple MacBook. Note that we have only a Matlab implementation of FAQ, whereas the

Table 2: Comparison of `FAQ` with optimal objective function value and the best result on the undirected benchmarks. Note that `FAQ` restarted 100 times finds the optimal objective function value in 3 of 16 benchmarks, and that `FAQ` restarted 3 times finds a minimum better than the previous state-of-the-art on all 16 particularly difficult benchmarks.

#	Problem	Optimal	$\texttt{FAQ}_{100}$	$\texttt{FAQ}_{3}$	previous min
1	chr12c	11156	**12176**	13072	13072
2	chr15a	9896	**9896***	17272	19086
3	chr15c	9504	**10960**	14274	16206
4	chr20b	2298	**2786**	3068	3068
5	chr22b	6194	**7218**	7876	8482
6	esc16b	292	**292***	294	296
7	rou12	235528	**235528***	238134	253684
8	rou15	354210	**356654**	371458	371458
9	rou20	725522	**730614**	743884	743884
10	tai10a	135028	**135828**	148970	152534
11	tai15a	388214	**391522**	397376	397376
12	tai17a	491812	**496598**	511574	529134
13	tai20a	703482	**711840**	721540	734276
14	tai30a	1818146	**1844636**	1890738	1894640
15	tai35a	2422002	**2454292**	2460940	2460940
16	tai40a	3139370	**3187738**	3194826	3227612

other algorithms are implemented in C. Nonetheless, `FAQ` runs nearly as quickly as both `U` and `QCV`, and significantly faster than `PATH`.

To investigate the performance of `FAQ` on undirected graphs, we ran `FAQ` on binarized symmeterized versions of the graphs ($A_{ij} = 1$ if and only if $A_{ij} \geq 1$ or $A_{ji} \geq 1$). The resulting errors are nearly identical to those presented in Figure 5, although speed increased by greater than a factor of two. Note that the number of vertices in this brain-graph matching problem—279—is larger than the largest of the 137 benchmarks used above.

6 Discussion

This work presents a fast approximate quadratic assignment problem algorithm called `FAQ` for approximately solving large graph matching problems, motivated by brain-graphs. Our key insight was to relax the binary constraint of QAP to its continuous and non-negative counterpart—the doubly stochastic matrix—which is the convex hull of the original feasible region. Numerically, we demonstrated that not only is `FAQ` cubic in time, but also its leading constants are quite small—10^{-9}—suggesting that it can be used for graphs with hundreds or thousands of vertices. Moreover, it achieves better accuracy than previous state-of-the-art approximate algorithms on on over 93% of the 137 QAPLIB benchmarks, including both directed and undirected graph matching problems. Because rQAP is non-convex, we also consider multiple restarts, and achieve improved performance for the particularly difficult benchmarks using only two or three restarts. We then demonstrate that the solution to our relaxed optimization problem, rQAP, is identical to that for QAP whenever the two graphs are simple and isomorphic to one another. Finally, we used it to match C. elegans connectomes to permuted versions of themselves. Of the four state-of-the-art algorithms considered, `FAQ` achieved perfect performance 100% of the time, whereas none of the other three algorithms ever

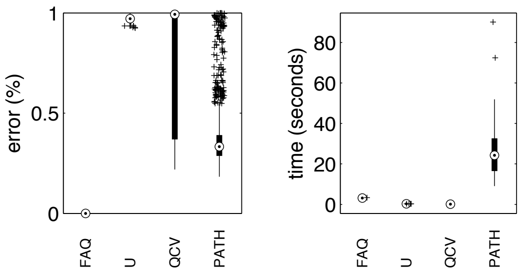

Figure 5: Performance of `U`, `QCV`, `PATH`, and `FAQ` on synthetic C. elegans connectome data, that is, graph matching the true connectomes with permuted versions of themselves. Error is the fraction of vertices correctly matched. Circle indicates the median, thick black bars indicate the quartiles, thin black lines indicate extreme but non-outlier points, and plus signs are outliers. The left panel indicate error (fraction of misassigned vertices), and the right panel indicates wall time on a 2.2 GHz Apple MacBook. `FAQ` always obtained the optimal solution, whereas none of the other algorithms ever found the optimal. `FAQ` also ran very quickly, nearly as quickly as `U` and `QCV`, and much faster than `PATH`, even though the `FAQ` implementation is in Matlab, and the others are in C.

achieved perfect performance. Moreover, `FAQ` ran about as fast as two of them, and significantly faster than `PATH`, even though `FAQ` is implemented in Matlab, and the others are implemented in C. Note that these connectomes have 279 vertices, more vertices than even the largest benchmarks.

Fortunately, our work is not done. Even with very small leading constants for this algorithm, as n increases, the computational burden gets quite high. For example, extrapolating the curve of Figure 1, this algorithm would take about 20 years to finish (on a standard laptop from 2011) when $n = 100,000$. We hope to be able to approximately solve rQAP on graphs much larger than that, given that the number of neurons even in a fly brain, for example, is $\approx 250,000$. More efficient algorithms and/or implementations are required for such massive graph matching. Although a few other state-of-the-art algorithms were more efficient than `FAQ`, their accuracy was significantly worse. So the search continues to find approximate graph matching algorithms with scaling rules like `QCV`, `U` or `RANK`, but performance like `FAQ`.

Additional future work might generalize `FAQ` in a number of ways. First, many (brain-) graphs of interest will be errorfully observed [36], that is, vertices might be missing and putative edges might exhibit both false positives and negatives. Explicitly dealing with this error source is both theoretically and practically of interest [12]. Second, for many brain-graph matching problems, the number of vertices will not be the same across the brains. Recent work from [26, 37] and [38] suggest that extensions in this direction would be both relatively straightforward and effective. Third, the most "costly" subroutine is LAP. Fortunately, LAP is a quadratic optimization problem with linear constraints. A number of parallelized optimization strategies could therefore potentially be brought to bear on this problem [39]. Fourth, our matrices have certain special properties, namely sparsity, which makes more efficient algorithms (such as "active set" algorithms) readily available for further speed increases. Fifth, for brain-graphs, we have some prior information that could easily be incorporated in the form of vertex attributes. For example, position in the brain, cell type, etc., could be used to measure "dissimilarity" between vertices. Finally, although this approach natively operates on both unweighted and weighted graphs, multi-graphs are a possible extension.

In conclusion, this manuscript has presented an algorithm for approximately solving the quadratic assignment problem that is fast, effective, and easily generalizable. Yet, the $\mathcal{O}(n^3)$ complexity remains too slow to solve many problems of interest. To facilitate further development and applications, all the code and data used in this manuscript is available from the first author's website, `http://jovo.me`.

A Linear Assignment Problems

The standard way of writing a Linear Assignment Problem (LAP) is

$$\text{(LAP)} \quad \underset{\pi}{\text{minimize}} \sum_{u,v \in [n]} a_{u\pi(v)} b_{ij} \tag{11a}$$

$$\text{subject to } \pi \in \Pi, \tag{11b}$$

which can be written equivalently in a number of ways using the notion of permutation matrix introduced in the main text:

$$\underset{P \in \mathcal{P}}{\text{argmin}} \, \|PA - B\|_F = \tag{12a}$$

$$\underset{P \in \mathcal{P}}{\text{argmin}} \, tr(PA - B)^{\mathsf{T}}(PA - B) = \tag{12b}$$

$$\underset{P \in \mathcal{P}}{\text{argmin}} \, -tr(PAB^{\mathsf{T}}) = \underset{P \in \mathcal{P}}{\text{argmin}} \, -\langle P^{\mathsf{T}}, AB^{\mathsf{T}} \rangle = \tag{12c}$$

$$\underset{P \in \mathcal{P}}{\text{argmin}} \, -\langle P, AB^{\mathsf{T}} \rangle, \tag{12d}$$

where $\langle \cdot, \cdot \rangle$ is the usual Euclidean inner product, i.e., $\langle X, Y \rangle \triangleq tr(X^{\mathsf{T}} Y) = \sum_{ij} x_{ij} y_{ij}$. While the objective function and the first two constraints of LAP are linear, the binary constraints make solving even this problem computationally tricky. Nonetheless, in the last several decades, there has been much progress in accelerating algorithms for solving LAPs, starting with exponential time, all the way down to $\mathcal{O}(n^3)$ for general LAPs, and even faster for certain special cases (e.g., sparse matrices) [21, 25].

That Eq. (6b) is a LAP is evident by considering Eq. (12d). If $A = \nabla_P^{(i)}$ and $B = I$ (the $n \times n$ identity matrix), then Eq. (6b) is identical to Eq. (12d).

To solve a LAP, consider a continuous relaxation of LAP, specifically, relaxing the permutation matrix constraint to a doubly stochastic matrix constraint:

$$\text{(rLAP)} \quad \underset{P}{\text{minimize}} \quad -\langle P, AB^{\mathsf{T}} \rangle \tag{13a}$$

$$\text{subject to} \quad P \in \mathcal{D}. \tag{13b}$$

As it turns out, solving rLAP is equivalent to solving LAP.

Proposition 1. *LAP and rLAP are equivalent, meaning that they have the same optimal objective function value.*

Proof. Although this proposition is typically proven by invoking total unimodularity, we present a simpler proof here. Let P' be a solution to LAP and let $P = \sum_{i \in [k]} \alpha_i P^{(i)}$ be a solution to rLAP for some positive integer k, permutation matrices $\{P^{(i)}\}_{i \in [k]}$, and positive real numbers $\{\alpha_i\}_{i \in [k]}$ such that $\sum_{i \in [k]} \alpha_i = 1$. Note that

$$\begin{aligned}\langle P, AB^{\mathsf{T}} \rangle = \langle \sum_{i \in [k]} \alpha_i P^{(i)}, AB^{\mathsf{T}} \rangle &= \sum_{i \in [k]} \alpha_i \langle P^{(i)}, AB^{\mathsf{T}} \rangle \\ &\leq \sum_{i \in [k]} \alpha_i \langle P', AB^{\mathsf{T}} \rangle = \langle P', AB^{\mathsf{T}} \rangle \leq \langle P, AB^{\mathsf{T}} \rangle,\end{aligned}$$

because P' is feasible in rLAP. □

This relaxation motivates our approach to approximating QAP.

Acknowledgment

The authors would like to acknowledge two helpful reviewers as well as Lav Varshney for providing the data. This work was partially supported by the Research Program in Applied Neuroscience.

References

[1] D Conte, P Foggia, C Sansone, and M Vento. THIRTY YEARS OF GRAPH MATCHING IN PATTERN RECOGNITION. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[2] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[3] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. QAPLIB A Quadratic Assignment Problem Library. *Journal of Global Optimization*, 10(4):391–403, 1997.

[4] Olaf Sporns, Giulio Tononi, and R Kotter. The Human Connectome: A Structural Description of the Human Brain. *PLoS Computational Biology*, 1(4):e42, 2005.

[5] Patric Hagmann. *From diffusion MRI to brain connectomics*. PhD thesis, Institut de traitement des signaux, 2005.

[6] Xi-nian N Zuo, Ross Ehmke, Maarten Mennes, Davide Imperati, Francisco Xavier Castellanos, Olaf Sporns, and Michael Peter Milham. Network Centrality in the Human Functional Connectome. *Cerebral cortex (New York, N.Y. : 1991)*, pages bhr269–, October 2011.

[7] John G Csernansky, Lei Wang, Sarang C Joshi, J Tilak Ratnanather, and Michael I Miller. Computational anatomy and neuropsychiatric disease: probabilistic assessment of variation and statistical inference of group difference, hemispheric asymmetry, and time-dependent change. *NeuroImage*, 23 Suppl 1:S56–68, January 2004.

[8] Marek Kubicki, Robert McCarley, Carl-Fredrik Westin, Hae-Jeong Park, Stephan Maier, Ron Kikinis, Ferenc A Jolesz, and Martha E Shenton. A review of diffusion tensor imaging studies in schizophrenia. *Journal of Psychiatric Research*, 41(1-2):15–30, 2007.

[9] Vince D. Calhoun, Jing Sui, Kent Kiehl, Jessica A Turner, Elena a Allen, and Godfrey Pearlson. Exploring the psychosis functional connectome: aberrant intrinsic networks in schizophrenia and bipolar disorder. *Frontiers in psychiatry / Frontiers Research Foundation*, 2:75, January 2011.

[10] Alex Fornito and Edward T. Bullmore. Connectomic intermediate phenotypes for psychiatric disorders. *Frontiers in psychiatry / Frontiers Research Foundation*, 3:32, January 2012.

[11] Alex Fornito, Andrew Zalesky, Christos Pantelis, and Edward T. Bullmore. Schizophrenia, neuroimaging and connectomics. *NeuroImage*, February 2012.

[12] Joshua T Vogelstein and Carey E Priebe. Shuffled Graph Classification: Theory and Connectome Applications. *Submitted to IEEE PAMI*, 2011.

[13] Robert P W Duin, Elbieta Pkalska, and Elbieta Pkalskab. The dissimilarity space: Bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, in press(April), May 2011.

[14] Scott Fortin. The Graph Isomorphism Problem. *Technical Report, University of Alberta, Dept of CS*, 1996.

[15] László Babali, Paul Erds, and Stanley M Selkow. RANDOM GRAPH ISOMORPHISM. *SIAM Journal on Computing*, 9(August):628–635, 1980.

[16] László Babali. Moderately Exponential Bound for Graph Isomorphism. *Fundamentals of Computation Theory*, pages 34–50, August 1981.

[17] Jianer Chen. A Linear-Time Algorithm for Isomorphism of Graphs of Bounded Average Genus. *SIAM Journal on Discrete Mathematics*, 7(4):614, November 1994.

[18] M Frank and P Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[19] Stephen P Bradley, Arnoldo C Hax, and Thomas L Magnanti. *Applied Mathematical Programming*. Addison-Wesley, 1977.

[20] Kurt M. Anstreicher. Recent advances in the solution of quadratic assignment. *SIAM Journal on Optimization*, 97(1-2):27–42, 2003.

[21] Rainer E Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. SIAM, 2009.

[22] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, March 1955.

[23] Dénes Knig. Gráfok és Mátrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[24] Jen Egeváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38:16–28, 1931.

[25] R Jonker and A Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[26] Mikhail Zaslavskiy, Francis R Bach, and Jean-philippe P Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

[27] Christian Schellewald, Stefan Roth, and Christoph Schnörr. Evaluation of Convex Optimization Techniques for the Weighted Graph-Matching Problem in Computer Vision. In *Proceedings of the 23rd DAGMSymposium on Pattern Recognition*, pages 361–368. Springer-Verlag, 2001.

[28] Rohit Singh, Jinbo Xu, and Bonnie Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. *RESEARCH IN COMPUTATIONAL MOLECULAR BIOLOGY*, 4453:16–31, April 2007.

[29] Shinji Umeyama. An Eigendecomposition Approach to Weighted Graph Matching Problems. *Analysis*, I(5), 1988.

[30] Zhi-Yong Liu, Hong Qiao, and Lei Xu. An Extended Path Following Algorithm for Graph Matching Problem. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1451–1456, January 2012.

[31] H A Almohamad and S O Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):522–525, 1993.

[32] Kurt M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, pages 471–484, 2009.

[33] R Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[34] J White, E Southgate, J. Nichol Thomson, and S Brenner. The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of Royal Society London. Series B, Biological Sciences*, 314(1165):1–340, 1986.

[35] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, Dmitri B. Chklovskii, Cold Spring, and Janelia Farm. Structural Properties of the Caenorhabditis elegans Neuronal Network. *PLoS Computational Biology*, 7(2):1–41, February 2011.

[36] Carey E Priebe, Joshua T Vogelstein, and Davi D Bock. Optimizing the quantity/quality trade-off in connectome inference. *Communications in Statistics Theory and Methods*, page 7, 2011.

[37] Mikhail Zaslavskiy, Francis R Bach, and Jean-philippe P Vert. Many-to-Many Graph Matching: A Continuous Relaxation Approach. *Machine Learning and Knowledge Discovery in Databases*, 6323:515–530, 2010.

[38] Francisco Escolano, Edwin Hancock, and Miguel Lozano. Graph Matching through Entropic Manifold Alignment. *Computer Vision and Pattern Recognition*, 2011.

[39] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Foundations and Trends in Machine Learning. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.